APPENDIX A

Informia: a Mediator for Integrated Access to Heterogeneous Information Sources

Maria Luisa Barja Tore Bratvold Jussi Myllymaki Gabriele Sonnenberger

UBS AG, Ubilab IT Laboratory
Bahnhofstrasse 45, CH-8098 Zurich
Switzerland
http://informia.ubilab.ch

Abstract

Integrated access to heterogeneous information sources is becoming increasingly important, largely due to the propagation of the World Wide Web and other distributed systems. Many existing solutions for dealing with heterogeneity are based on a database-style modeling of data, meta-information, and queries. Others focus on information retrieval (IR) techniques such as best-match queries, relevance assessment, and content-based source selection. Not much research exists which combines the two approaches.

In this paper, we present a system named Informia which combines techniques from the database and IR fields and allows the user to access a large and diverse set of information sources. We discuss user requirements for information seeking in a heterogeneous environment, derive system features necessary to support the users' needs, and show how Informia supports these features. We also describe Informia's extensible and customisable architecture, which facilitates the development of specialised applications for particular problem domains.

Keywords: Networked Information Retrieval, Database Systems, Mediators, Heterogeneous Information Access.

1 Introduction

Two trends in information technology are rapidly transforming the way many organisations process data. On the one hand, the expansive networking of computer systems has brought together workstations and mainframes to form vast collections of distributed, online information services. The prime example and manifestation of this trend is the World Wide Web (WWW). On the other hand, information units "published" in these networks are no longer homogeneous, but may be compositions of heterogeneous data types, e.g. text documents, images, and database records.

In an attempt to harness the power of these distributed, heterogeneous systems into the hands of an end-user, systems have been built which, at least technically, allow the user to tap into any one of an array of information sources (consider the WWW, for instance). These solutions do not really make information seeking much easier or simpler, however, because they require the user to learn the intricacies of a large number of interfaces, system features, and data formats. Thus, users are still expected to know how to receive, manipulate, and combine ever more complex types of data from ever more sources.

One approach for reducing the complexity is to connect several systems to a mediator [14] which integrates disparate data items dynamically and presents them to the user through one interface or access point. To perform such integration automatically requires rich meta-information about documents and sources providing the documents. The typical system flow in a mediator architecture, then, is for the mediator to use *source content meta-information* to identify sources potentially contributing to a user's information need, then split and translate the user's query according to *structural meta-information*, next forward sub-queries to the sources and receive results, and finally consolidate the results.

Several different approaches have been taken by the research community for splitting queries and integrating results. Database-oriented systems typically use the structural information of schemas for this purpose [1, 2, 7, 10]. Query distribution and results integration may also be determined by a mediator with embedded domain knowledge [8] or by considering the computational capabilities of sources [5]. Knowledge exchange languages and semantic agents have also been used [3, 9].

With the exception of the notion of *graded sets* in Garlic [5], these systems place little emphasis on relevance measures and other information retrieval (IR) techniques such as relevance feedback. The MeDoc Information Brokering System [4] makes advances in this direction but is targeted at structured, bibliographic data and is not extensible to other subject domains. WWW meta search engines [13] merge relevance scores from different sources and operate independently of the domain. However, the engines are restricted to a specific document and query model: HTML documents and full-text queries.

Perhaps because of the historical divide between database and IR communities and their disjoint research agendas, not much research exists which combines best-match queries and relevance measures with a database-style modeling of data, meta-information, and queries. In this paper, we identify requirements for information seeking in a heterogeneous environment, and present the Informia system, the most salient features of which are:

CIKM 98 Bethesda MD USA

- it provides access to a heterogeneous and dynamic set of sources, e.g. database systems, IR systems, document management systems, WWW search engines, electronic mail folders, and file systems;
- it combines formal models as found in database systems with a consistent relevance assessment of documents retrieved from different information sources;
- it is based on an extensible object-oriented framework which lends itself well to building customised applications for particular problem domains.

Informia is in operational use within our organisation. A public version of the system is accessible on the WWW at `http://informia.ubilab.ch`.

We have structured the paper as follows. In Section 2, we identify the requirements of a user who wishes to seek information from heterogeneous sources. The resulting system requirements are compared with the capabilities of previously proposed systems in Section 3. In Section 4, we describe the Informia system and its main components. Implementation details are described in Section 5. We present our conclusions and future plans in Section 6.

2 Requirements for a Heterogeneous Information System

Consider a user who wishes to use a heterogeneous information system for retrieving information from any available source that might satisfy his or her particular information need. The user is most often concerned about the relevance and timeliness of information, not about which source the information comes from, which data model it adheres to, or which query interface was used to retrieve the information. The user should not need to be aware of the technical underpinnings of the system, nor be limited as to what type of information he or she can access or from what type of sources. Ideally, there should be a *uniform interface* for expressing queries for multiple information types and a single, *consolidated set* of results *consistently ordered* regardless of the particular scoring mechanism used by each source. The user should also be allowed to select, compare, cluster, and otherwise analyse information sources at a meta-level.

To support these user requirements, a system for accessing heterogeneous information sources needs mechanisms for hiding the differences between sources, for identifying sources likely to contain relevant information, and for combining results. System *scalability*, *extensibility* and *customisability* are also important because the system needs to adapt as the environment or the user's needs change. Scalability is essential for handling the exponentially growing number of available information sources, as is the case on the WWW. No single WWW crawler indexes the entire WWW and no single corporate database or search engine provides one-stop access to global information.

Extensibility is critical because new information sources and new interfaces, protocols, and formats emerge constantly. Users and system administrators should be able to extend the system with minimum manual effort. Ideally, the system should be able to discover new information sources automatically and figure out how to talk to them. A system should also be able to adapt to a changing environment because information sources, especially external ones, typically enjoy autonomy and may change unexpectedly. A heterogeneous system should therefore not depend on other systems' data models, query languages and vocabularies, or the protocols and interfaces they support. Internally, a heterogeneous system therefore needs to employ a data model and query language that are rich enough to subsume the data and query representation capabilities of other systems, and it should provide a flexible abstraction mechanism which hides the interface details of an information source.

Customisability is important because it is unrealistic to expect a single generic system to be able to handle all application domains and information seeking tasks as competently as a tailored system. Therefore, an architecture in which individual components can be customised to fit a particular application domain is desirable. A customised results integration component, for instance, can embed knowledge about domain-specific vocabularies, transformation rules, or similarity measures to improve retrieval effectiveness.

Much of what we have described requires *rich meta-information* because the fewer *a priori* assumptions we make about information sources, the fewer rules we can hardwire in a system and the more meta-information the system needs for making decisions at run-time. Meta-information is critical to the success of an extensible heterogeneous system and should preferably be extracted automatically. Since the quantity and quality of meta-information that can be automatically extracted from a source vary, the system should be able to cope with missing or incomplete meta-information as well. Ideally, meta-information about a source would include all of the following: the identity of a source (e.g. its name, type and location); service parameters (e.g. cost, response time, reliability, availability, etc.); content descriptions (e.g. major topics covered in a text collection); retrieval functionality supported (e.g. truncation patterns and stopword lists); and schema information, i.e. the structure of the information types supported by a source (e.g. names of information structures, names and types of attributes, etc.).

Schema information has particularly important uses in a heterogeneous information system. First, it guides the system in merging structured, disparate pieces of information. Second, it allows document retrieval to be more targeted and document attributes to be matched with query conditions according to their type. For example, the string "April" is closer to "March" than to "August" in a date field. Third, schema information can provide valuable input for automatic source selection.

3 Related Work

We now look at some of the heterogeneous information systems proposed in the literature and compare them with the requirements highlighted in the previous section. Common to many of these systems is that they can be described using a three-tier mediator architecture [14] shown in Figure 1. In this model, a user or application communicates with a mediator (sometimes referred to as a broker or agent) which retrieves and consolidates information from multiple sources. The mediator interacts with information sources via wrappers (also called proxies, adapters, or converters) which constitute an abstraction mechanism and thus contribute to the extensibility of the system.

WWW meta search engines like MetaCrawler[1] can be viewed as information integration tools since they distribute a user's query to multiple individual search engines and consolidate the results by removing duplicate result items and normalising the relevance scores. FUSION-II[2] [13] also

[1] `www.metacrawler.com`

[2] `lorca.compapp.dcu.ie/fusion2`

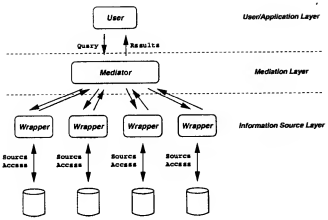

Figure 1: Three-tier mediator architecture of a heterogeneous information system.

provides relevance feedback and query expansion capabilities. These systems, however, are restricted to a fixed set of WWW sources and do relatively little to model documents or queries in a more general manner. Other meta search engines are more limited in terms of results integration. SavvySearch[3], Inference Find[4], and MetaFind[5] consolidate results but do not display an explicit relevance score to the user. All-4-One[6] shows the results obtained from different sources in separate lists and leaves the task of results consolidation to the user.

Outside the realm of WWW meta search engines, few heterogeneous retrieval systems aim to provide a consistent notion of relevance across different information sources. The MeDoc Information Brokering System [4] is one such system but its scope is limited to full-text databases containing highly structured bibliographic information. The system also assumes that retrieval status values (RSVs) obtained from sources are compatible and can be used without normalisation as the basis for combining ranked lists.

Several heterogeneous information systems are described in the database literature, including TSIMMIS [8], Garlic [5], SIMS [1], InfoBus [2], and Infomaster [9]. Common to these systems is that they are general in scope and are extensible to a variety of sources (database systems, WWW, file systems, etc.) via a wrapper mechanism. Information Manifold [10] accesses WWW sources only but is extensible via declarative source descriptions.

The systems use different formalisms for representing source content and structure. For instance, TSIMMIS uses an Object Exchange Model (OEM) in which objects are self-describing, i.e. each object is associated with its description, and there is no explicit schema. Attribute models (InfoBus), knowledge interchange formats (Infomaster and InfoSleuth [3]), object models (SIMS and Garlic), and object-relational models (Information Manifold) have also been explored.

The system most similar to our Informia system is Garlic in that both are based on the ODMG data model. Whereas other systems are oriented towards database-style Boolean queries and perform no relevance assessment of results, Garlic allows the combination of exact and approximate query match semantics, as does Informia. However, while Garlic represents the relevance of results via graded sets, Informia takes a more direct approach and explicitly supports non-Boolean relevance assessment.

Source selection in these systems is either based on structural information alone (InfoMaster, Garlic, and SIMS), or on a combination of structural and content descriptions (InfoBus and Information Manifold). TSIMMIS allows mediators to embed domain-specific knowledge about how to distribute queries across sources. Informia combines these approaches: it allows structural and content information to be used by a source selector component, which may also be customised to provide additional domain-specific rules and hints.

One of our key requirements is to allow specialised retrieval and integration applications to be developed. A first step towards this is to split system functionality into reusable and customisable components. Existing approaches include using networked agents (InfoSleuth), services as distributed objects (InfoBus), APIs (Garlic) or semantic descriptions of domains and sources (SIMS). Our approach is that of using an object-oriented application framework [11] with a set of communicating and cooperating components. This allows customisation, extension, and reuse both of components and of the architecture as a whole.

4 Informia

4.1 Overview

Following up on the requirements outlined in Section 2, we now describe Informia, a mediator system for retrieving and integrating information from heterogeneous and distributed information sources. The goals of Informia are threefold: to provide abstraction from information sources, to support information retrieval and integration tasks, and to be scalable, extensible, and customisable. These are achieved by combining database and IR techniques, and embedding them in an extensible object-oriented framework architecture.

Informia's internal data model and query language (see Section 4.2), its *common access interface (CAI)*, and wrappers (see Section 4.5) provide abstraction from locations of information sources and from their different query languages, access interfaces, data models, and schemas. This gives a uniform view of the entire "information space", across different types of information sources. Thus, to the internals of Informia, all information sources appear to have the same data model and query language (i.e. Informia's). Consequently, Informia is open-ended with respect to the number and types of information sources it can access, despite differences in their data models and query languages.

The described abstraction mechanisms allow different types of information objects and information sources to be treated uniformly, which is essential for supporting higher level retrieval and integration functions effectively. Informia supports a number of such functions, such as browsing of source meta-information in its *meta-information repository (MIR)* (see Section 4.4), transformation of queries, and combination of retrieval results (see Section 4.6).

As argued in Section 2, no "silver-bullet one-size-fits-all" tool is likely to be adequate for all user tasks, information needs, and problem domains. Informia's architecture (see Section 4.3) is a component-based object-oriented framework [11], which is highly customisable and extensible. Custom components can be integrated with existing functionality to produce specialised applications. For example, when seeking for information we use Informia as a

[3] guaraldi.cs.colostate.edu:2000
[4] www.inference.com/infind
[5] www.metafind.com
[6] www.all4one.com

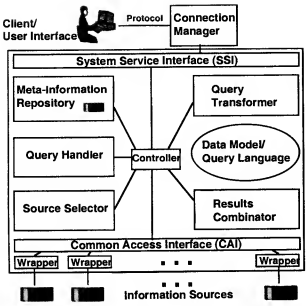

Figure 2: Informia's internal architecture.

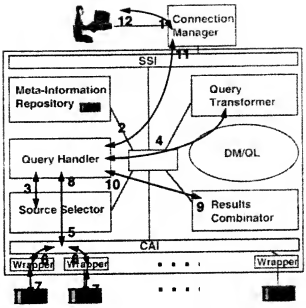

Figure 3: Query processing example.

meta-search engine through a web browser, but it can also be used as middleware in specialised information processing applications. In the design of Informia, we have aimed for a mediator *architecture*, which can be adapted and used in a variety of contexts, rather than a single static system.

4.2 Data Model and Query Language

Informia's data model is based on the Object Database Management Group ODMG 2.0 object database standard [6]. ODMG provides modeling of information objects such as integers, strings, lists, and enumerations, as well as of meta-objects (or schemas), i.e. descriptive information about the structure and composition of information objects. A schema definition in Informia is used as a means for providing a generalised, uniform and a non-materialised view of the information stored in the underlying sources. This is paramount for achieving internal homogeneity as the basis for information processing and analysis components such as the MIR, the query transformer, the source selector, and the results combinator.

ODMG is both comprehensive and general, but we have still made a few minor extensions for adequately supporting the needs of Informia. Among the extensions are explicit support for relations (tuple-homogeneous sets) and multimedia types (such as text, image, audio and video) as well as for references (e.g. URLs) to non-local objects.

For the internal representation, transformation and processing of queries, we choose OQL (Object Query Language). OQL is an expressive language that allows queries addressing complex structured objects using a *select ... from ... where ...* syntax. This covers and can be translated into queries for a wide range of sources. In particular, querying OODBMSs or SQL-based information sources is straightforward. Traditional full-text IR queries are represented using an artificial attribute *qstring* for the query string in the *where* clause, e.g. *qstring contains "stock price developments and bank merger announcement"*. Our extensions to OQL are aimed at querying sources containing semi-structured and unstructured information, using additional operators and weighted query conditions. Our user interface allows the user to express queries as free text as well as in this language. At present we allow expert users to use the query language directly, while still offering a template based interface where a query form is automatically transformed into an internal query language representation.

4.3 Architecture

Informia's internal architecture, depicted in Figure 2, follows the classical mediator model shown in Figure 1. It can be roughly separated into three parts: interaction with information sources (bottom of figure), interaction with clients and user interfaces (top), and Informia's internals and components (middle).

Each information source is abstracted by a wrapper which provides a mapping between the query language of Informia and the query language of an information source. Correspondingly, it also provides a mapping between the data model of an information source and the data model of Informia. Wrappers are controlled by the common access interface (CAI). All interaction between Informia and the information sources take place through the CAI. At the top of the architecture, a *connection manager* component handles interaction between Informia and different user interfaces or client communication protocols. Our present connection manager supports a web browser user interface via HTTP/CGI, as well as lower level plain text TCP/IP and console access. All access to Informia goes via a *system service interface (SSI)* which makes the services provided by the other components available to external systems.

The components inside of Informia are responsible for different aspects of query processing and retrieval. The meta-information repository (MIR) (see Section 4.4) stores schemas and other information about the available information sources. The *source selector* can use the MIR to identify information sources likely to contain information relevant to a query. Informia's *query transformer* transforms a query into sub-queries; one sub-query for each information source to be accessed. The query transformer makes use of

information in the MIR regarding schemas of information sources and the relations between them. The consolidation of result sets returned from different information sources is the task of Informia's *results combinator* (see Section 4.6). The *query handler* coordinates the processing of a query by deciding which of the other components to contact, which of their functions to invoke, and in what order.

A key design requirement of Informia is to support the development of specialised retrieval and integration applications. Informia is implemented as an object-oriented application framework [11] with a set of communicating and cooperating components. Components are modeled as abstract classes that can be specialised by deriving concrete subclasses. The connection manager, query handler, source selector, query transformer, results combinator, as well as the MIR and CAI are all components offering a range of services through their interfaces. A *controller* object serves the administrative role of keeping track of the other components and the system parameters. Informia comes with a set of default "core" components providing basic functionality. By specialising individual components, Informia can be customised for particular applications. Each component knows only of the interfaces of the other components, not of their implementation, and so one or more components can be replaced by specialised versions conforming to the same interface without having to change other parts of the system. Further customisation is possible by embedding and making use of some or all of Informia's components directly from other applications, or by adding new components. All components have in common that they operate on Informia's data model and query language.

The role of the different components is perhaps best illustrated by an example, see Figure 3. Consider the query *select * from * where author="Cheswick" and title contains "firewalls"*, posed to Informia's SSI through the connection manager (1). The query is immediately passed to the query handler (2), which is in charge of planning and controlling the execution of the query. Informia's "core" query handler passes the query directly to the source selector (3) which, using the MIR, identifies a set of promising sources, e.g. catalogues of bookstores or libraries with schemas containing author or title attributes. Then the query transformer (4), again using the MIR, transforms the query into a set of sub-queries, one for each information source. These may be *select * from "Amazon.com" where author="Cheswick"and title contains "firewalls"* and *select * from "UniLibrary" where bookauthor="Cheswick" and booktitle contains "firewalls"*, etc. The set of sub-queries is then forwarded to the CAI (5), which passes each sub-query to the appropriate wrapper (6) for the respective sources. Wrappers run concurrently monitored by the CAI. Each wrapper translates its sub-query into a query for its information source, submits it (7), waits for the results, and translates them into Informia's data model. The query handler receives the result sets (one set from each source) from the CAI (8), and passes them on to the results combinator (9). Results are then merged, before finally being passed back (10,11,12) to the user through the SSI. Although this example illustrates a relatively typical scenario, components can in principle be used, embedded, extended, and invoked in a number of ways.

4.4 Meta-Information Repository

The MIR is responsible for managing and analysing meta-information about information sources, including content

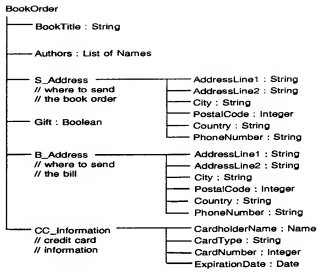

Figure 4: Schema example.

and schema information. Here we will elaborate on schema information. Schema information is provided to the MIR by the wrappers, and is represented as meta-objects in ODMG 2.0. A schema representation as well as its attributes are associated with a name and an optional textual description. Figure 4 gives a simple example illustrating the structure of a schema.

Automatic integration of schemas from different sources is very hard due to semantic and syntactic inconsistencies. Instead, we have taken the approach of applying IR-like indexing to schema information, using a best-match retrieval approach where the retrieval results are ordered by estimated relevance.

The MIR indexes all textual information in a schema. This includes the schema name, descriptive comments, attribute names, and attribute types. Indexing features consist of stemmed terms, plus a path expression identifying exactly the subpart of the schema description the feature was derived from. Path information is needed because Informia's query language supports both unstructured and structured queries, containing conditions like *BookOrder.B_Address.City = "Glasgow"*.

Compared with text, schema descriptions are much more terse. Artificial words, such as S_Address, occur frequently, which exacerbates the well-known word-mismatch problem. To increase recall, we apply different indexing methods in parallel. Currently, these are preservation of the original string, left-hand stripping guided by special characters (e.g. S_Address → Address), stemming (using Porter's algorithm), linguistic decomposition and determination of the compound head (e.g. ExpirationDate → Date) and translation of the normalised word form into WordNet synonym sets. The results produced by the different methods are stored in separate indexes.

The MIR offers different facilities for querying schema information. Queries can be either *path queries*, e.g. *BookOrder.B_Address.City*, which result in the retrieval of sources whose schemas contain similar paths, or alternatively in the retrieval of schema attributes reachable by a similar path. *Schema queries* take as input a (partially defined) schema and produce as result schemas which are similar to the given

Source	Type	Schema
ODBC database	database	extracted automatically from database
Altavista	WWW	title, summary, URL, size, date and language of document
Excite	WWW	relevance, title, URL and summary of document
Amazon.com	WWW	title, author, type (e.g. hardcover), price, year published, and remarks
Exchange Rates	WWW	currency pair, bid and ask quote, time of quote, day's high and low quote
CNN News	WWW	title, date, relevance, source and summary of news item
File system	local	filename, size, modification date, owner, group, permission flags and content
Email messages	local	folder name, sender, subject, date, recipients, CC list, length and content

Table 1: Examples of information sources and schemas.

one. Queries to the MIR are indexed in a similar way as described above, and are then matched against all indexes. If there is a match, most often more than one index will produce results, which then have to be combined into a single final result. In the current version of MIR, we apply a simple linear combination method where all intermediate results have the same weight, with RSVs being normalised beforehand.

Interlinking of meta-information is based on the results of the indexing. The MIR computes a similarity matrix by using each schema in turn as a query against the other schemas. Links between schemas are established according to the retrieval results. These links describe explicitly which parts of two schemas are considered to be similar and also specify the degree of similarity. These links, together with the MIR's ad-hoc schema querying facilities form the basis for directed browsing of schema information.

4.5 Common Access Interface

The task of the CAI is to hide the query interface and data model differences between information sources. The abstraction mechanism presents a uniform interface for shipping queries and retrieving results and meta-information, thereby simplifying the implementation and operation of Informia's other components. The translation of a Informia query to the native query language[7] of an information source and the translation of retrieved result items to the Informia data model are accomplished by a wrapper. The wrapper is also responsible for meta-information extraction, simple rescaling of relevance assessment scores, and management of the network connection and protocols.

The CAI provides an open-next-close interface for retrieving results from an information source. The CAI typically receives a set of queries from the query handler, which the CAI then forwards to the wrappers of the corresponding information sources. The query handler also provides a set of query parameters which specify the maximum number of results to retrieve from each source and the maximum amount of time allowed. Next, the query handler retrieves results iteratively from the CAI and terminates the query when no more results are available.

Informia currently supports three types of information sources, see Table 1. In the first group are database systems such as Oracle and any relational databases accessible through the Open Database Connectivity (ODBC) interface. Support for the ObjectStore and Shore object-oriented databases is being developed. Schema information is automatically extracted from these database sources and stored in the MIR. In addition to allowing Boolean query conditions on attributes, the database wrappers can filter out those database records or objects whose textual representation does not contain any of the terms of a free-text query.

The second group of information sources includes search engines and other information sources on the WWW. If the information source is a search engine, the wrapper translates the original query to a URL or an HTTP form, forwards it to the WWW site hosting the search engine, and decodes the result page to identify result items.

In the third group are information sources in the user's local file system, such as regular files and electronic mail folders. For these sources the wrapper itself traverses the files and calculates relevance scores on-the-fly.

4.6 Results Combinator

The task of the results combinator is to provide a consistent notion of relevance for information retrieved from different information sources. This is a hard problem, since information sources must generally be assumed to be independent. They therefore apply different ways of assessing relevance, including not providing an RSV at all. A document given, say, a score of 0.8 from source A cannot be assumed to be more relevant than a document given a score of 0.6 from source B[8].

In Informia, we tackle this problem by the following stages:

(1) Every retrieved document is associated with a *source relevance score* in the range [0..1] as given by the source. Some sources provide this score directly, while for others we apply normalisation (e.g. divide by a constant K for scores in the range $[0..K]$, or map non-numeric values (e.g. colour palettes) to a corresponding score in [0..1]). Where no score is available, a score is estimated from the rank (e.g. score=max((10-rank)/10 , 0.1)), and for objects/tuples returned from database systems the score is set to 1.0. For the remaining sources which produce no relevance score, rank, or other ordering, the source relevance score is set to 'unknown' and ignored. In cases where the same document is retrieved by several sources, e.g. the same URL retrieved by multiple search engines, the document is associated with multiple source relevance scores; one for each source.

(2) For every retrieved document, Informia calculates an *internal relevance score* based on a vector-space similarity between the textual representations of the query and the document. As the text of the document we use whatever is returned from the source, e.g. the first 20-40 words of the document commonly returned by web search engines, or the plain text representation of a relational database tuple.

[7] Some information sources, e.g. the file system, do not have a query-oriented interface. The wrapper in this case emulates a query interface and contains a simple query processor.

[8] Note that this applies even if source A and B use exactly the same retrieval engine, because relevance scores may be affected by the characteristics of document collections, e.g. different term weights may apply.

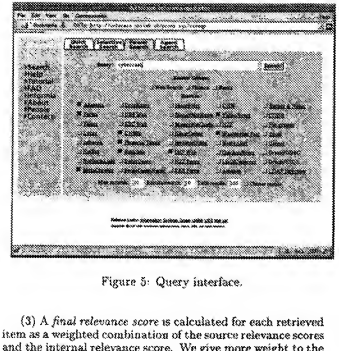

Figure 5: Query interface.

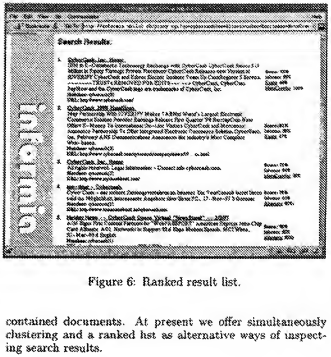

Figure 6: Ranked result list.

(3) A *final relevance score* is calculated for each retrieved item as a weighted combination of the source relevance scores and the internal relevance score. We give more weight to the internal score than the source scores, but increase the relative weight of the aggregate source score progressively when a document is retrieved by more than one source.

Retrieved documents are ranked in descending order based on the final relevance score. See Figure 6 for an example of how the final score (denoted "Score:") the internal score (denoted "Informia:"), and the scores from the sources are presented.

The information available at step 2 is often just a small part of the entire document, which detracts from the accuracy of the internal relevance score. A more accurate approach is to fetch each document and reassess its relevance fully[9], but this is costly in terms of network download time, thereby making it unattractive for interactive querying. Our approach is clearly a compromise. It is relatively fast, thus being suitable for an interactive retrieval environment, but it is presumably less accurate than reassessing each retrieved document in its entirety. We consider the approach as experimental, and we would like to tune it for improved performance. Note, however, that experiments in the traditional IR sense, using recall and precision as measures of retrieval effectiveness, cannot easily be applied here. A realistic testbed for our system would be very large, e.g. involving most of the WWW, which excludes comprehensive manual relevance assessments, and is also difficult to freeze for guaranteeing repeatability of results.

Informia's framework architecture allows full customisation of the result combination steps, from using different weighting and internal relevance assessment approaches to replacing parts or whole of the current implementation.

It should be noted that imposing a linear ordering upon results from different sources and on different topics is not the only approach to results combination, and not necessarily the best. As an alternative, we have integrated clustering tools [12] in Informia. The retrieved documents are grouped into a hierarchy of clusters using an agglomerative approach based on vector-space document similarities. Each cluster is associated with a descriptor of terms co-occurring in the contained documents. At present we offer simultaneously clustering and a ranked list as alternative ways of inspecting search results.

[9]Informia supports this in a non-interactive *agent* query mode.

5 Implementation Status

A prototype of Informia has been implemented and is in operational use within our organisation. One installation of the system retrieves information from Internet sources as well as from sources on our local, departmental network. Another installation is intended primarily for sources on our global corporate intranet (e.g. Lotus Notes databases) but it also provides access to Internet sources.

The user interface is HTML-based and runs in a Web browser. Queries can be formulated at different levels of parameterisation detail. An agent query mode, where the system performs a more comprehensive analysis of results in the background and sends the results back to the user via email, is also available. Client-server communication in Informia is handled via a Common Gateway Interface (CGI) over a TCP/IP connection. Informia can also be operated in command-line mode locally or through a plain TCP/IP connection. The server runs on Solaris and consists of roughly 70,000 lines of C++ code.

Informia currently provides access to more than 30 different WWW sources plus local sources such as the file system and ODBC databases. The WWW sources range from generic search engines to news providers to financial information sources, among others. The wrappers for these sources have been created manually, but we are currently deploying an analysis tool which extracts query parameters of a Web source and infers the structure (schema) of result pages. This process requires very little human input and allows Informia to use a single, generic wrapper for all WWW source access.

Source selection is currently manual, allowing the user to select sources or source groups from a panel (see Figure 5). The user can browse source meta-information before selecting sources, or instruct Informia to query all sources. We are currently experimenting with automatic source selection based on source content information, schema information, relevance feedback, and source availability and performance data.

The current implementation does not fully support structured queries or query weights. The query language has the necessary expressive power to represent these features, but the required parts of Informia's query transformation, query execution, and results combination modules have not been implemented yet.

The results combinator has been implemented with all features described in Section 4.6. The relevance of all result items is reassessed (see Figure 6) and combined with the relevance score indicated by the source (or multiple sources, if they return an identical result item). Hierarchical clustering of result documents has also been implemented.

6 Conclusions and Future Work

Heterogeneity of information and information sources makes effective information seeking hard. We have argued that database techniques such as having an expressive internal data model and query language, together with a meta-information repository and corresponding meta-information analysis techniques, constitute a necessary foundation for a mediator system. On the other hand, in order to alleviate the problem of information overload when results to a query are presented, it is paramount for these to be ranked according to consistent relevance assessments. Further, functionality for retrieval and integration of information must be supported by an easily extensible and customisable architecture for addressing particular problem domains.

We claim that the Informia architecture goes a long way towards meeting these requirements. Our data model, query language, wrappers, common access interface, and meta-information repository provide a scalable and extensible foundation for implementing retrieval and integration functionality. For instance, we have used this foundation effectively for implementing our results combinator and meta-information analysis tools. Customisability is achieved by adopting an object-oriented framework design which provides the basis for developing a wide range of specialised applications.

Although the Informia system is implemented and running, it is under constant development. Ongoing work is focusing on experimenting with: i) tools for automatic extraction of schema and content information directly from the sources, with the main goal of automatically generating wrappers; ii) adding further analysis techniques to the MIR, including use of source content information; iii) providing automatic source selection based on schema, content information, feedback information and performance data; iv) clustering as a means of organising both information in the MIR as well as search results. Eventually we would like to gain experience in developing specialised applications by applying Informia to information searching problems in the financial domain.

Acknowledgements

We thank M. Berney, P. Cho and S. Cordier, Ubilab, for their contribution to the development of Informia. D. Harper, M. Mechkour, and G. Muresan at Robert Gordon University proposed and developed the clustering framework of the system. K. Dittrich and R. Domenig at the University of Zurich provided initial input, and have taken part in the development of Informia's query language. Last but not least, we thank H.-P. Frei, Ubilab, for very valuable input at all stages of the project.

References

[1] Y. Arens and C. A. Knoblock. SIMS: Retrieving and integrating information from multiple sources. *SIGMOD Record*, 22(2):562–563, June 1993.

[2] M. Baldonado et al. Metadata for digital libraries: Architecture and design rationale. In *Proceedings of the ACM International Conference on Digital Libraries*, pages 47–56, Philadelphia, PA, July 1997.

[3] R. J. Bayardo, Jr. et al. The InfoSleuth project. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 543–545, Tucson, AZ, May 1997.

[4] D. Boles, M. Dreger, and K. Grossjohann. MeDoc Information Broker – Harnessing the information in literature and full text databases. In *Proceedings of the ACM SIGIR Workshop on Networked Information Retrieval*, Zurich, Switzerland, Aug. 1996.

[5] M. Carey et al. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proceedings of the International Workshop on Research Issues in Data Engineering (RIDE)*, pages 124–131, Taipei, Taiwan, Mar. 1995.

[6] R. Cattell et al., editors. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann Publishers, San Francisco, CA, May 1997.

[7] K. R. Dittrich. A Universal Query Service for heterogeneous information sources, Sept. 1996. Unpublished.

[8] H. Garcia-Molina et al. Integrating and accessing heterogeneous information sources in TSIMMIS. In *Proceedings of the AAAI Symposium on Information Gathering*, pages 61–64, Stanford, CA, Mar. 1995.

[9] M. R. Genesereth, A. M. Keller, and O. M. Duschka. Infomaster: An information integration system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 539–542, Tucson, AZ, May 1997.

[10] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *Journal of Intelligent Information Systems*, 5(2):121–143, Sept. 1995.

[11] T. Lewis et al. *Object Oriented Application Frameworks*. Manning Publications Co., 1995.

[12] M. Mechkour, D. J. Harper, and G. Muresan. The WebCluster project. Using clustering for mediating access to the World Wide Web. Poster, ACM SIGIR 1998.

[13] A. F. Smeaton and F. Crimmins. Relevance feedback and query expansion for searching the Web: A model for searching a digital library. In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries*, pages 99–112, Pisa, Italy, Sept. 1997.

[14] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.